Week-12

Aim: to implement a deep convolutional Generative Adversial network (DCGAN) of generating complex RGB colour image truth resemble real-world images

objective:

1, to understand the working of Generative Adversial network (GAN's)

2, to use convolutional layer for image generation and discrimination

3, to trach the Generator to produce redistic image and the discriminator to distinguish real from fake

4, to evaluate the quality of Generated images visually after training.

pseudo code:

Begin

load dataset of real colour images

→ Resize and normalize image to (-1,1]

Define Generator Network (G):

input : random wise rector (z)

layer: Series of conv Transpose 2D + BatchNo + Relu

output : RGB image.

Define Discriminator Network (D):

input : RGB image

layer : Series of conv2D + Batch Normal + leaky ReLU

output : single probability

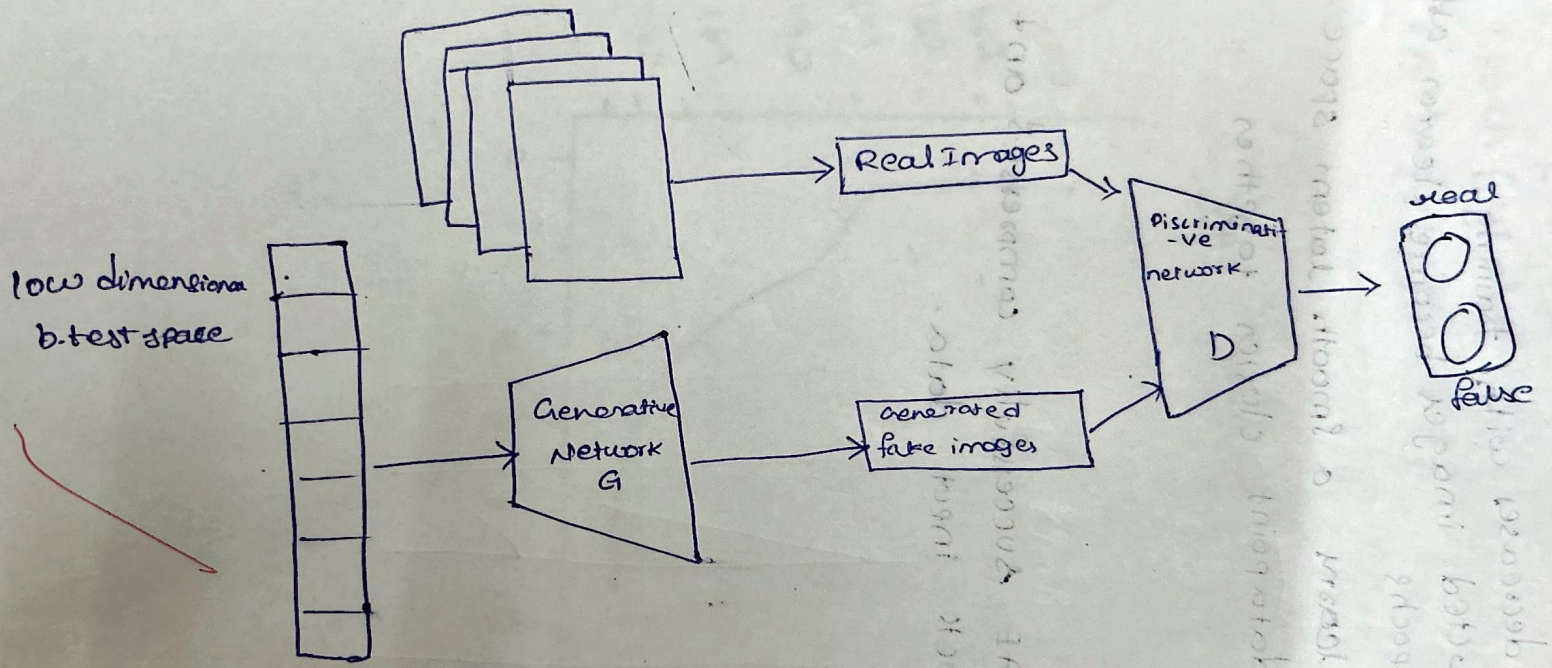
Seat loss function = Binary cross entropy

set optimizer = Adam ($r = 0.0002$, $B_1 = 0.5$)

for each epoch

Train discriminator

GAN Architecture For image



Train Generator

Repeat until generator produces realistic color images

END

Algorithm:

1) load a colour image dataset
2) Normalize pixel value to [-1,1]
3) Define generator using conv 2D Transpose layer
4) Define discriminator using conv 2D layers
5) combine both to from DCGAN
6) Train using adversarial loss - generator tries to fool discriminator
7) periodically generate and display fake images

observation:

→ the generator gradually learned to produce redistic colour images
→ the discriminator improved in distinguishing real and fake images
→ Generated images were visually similar to the training dataset after sufficient epochs
→ training demonstrated the adversarial learning process of GAN effectively

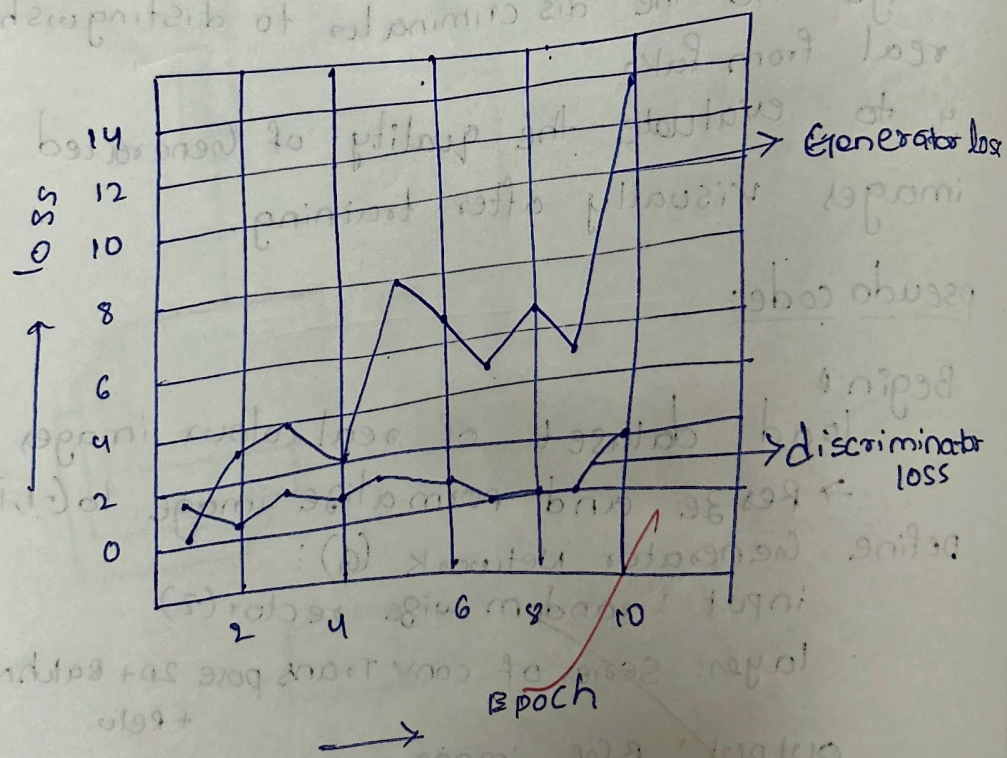
Result:

The DCGAN succesfully Generated realistic looking colour images, proving the effectiveness of adversarial learning.

Output :-

Epoch [1/10] D Loss : 1.0992 ; G loss : 0.5678
Epoch [2/10] D loss : 0.5060, G loss : 2.7113
Epoch [3/10] D loss : 1.2437, G loss : 3.9827
Epoch [4/10] D loss : 0.4229, G loss : 2.2684
Epoch [5/10] D loss : 1.1456, G loss : 8.1007
Epoch [6/10] D loss : 1.0081, G loss : 6.7625
Epoch [7/10] D loss : 0.0252, G loss : 4.5718
Epoch [8/10] D loss : 0.0043, G loss : 6.3371
Epoch [9/10] D loss : 0.1380, G loss : 4.7357
Epoch [10/10] D loss : 2.3294, G loss : 14.0363

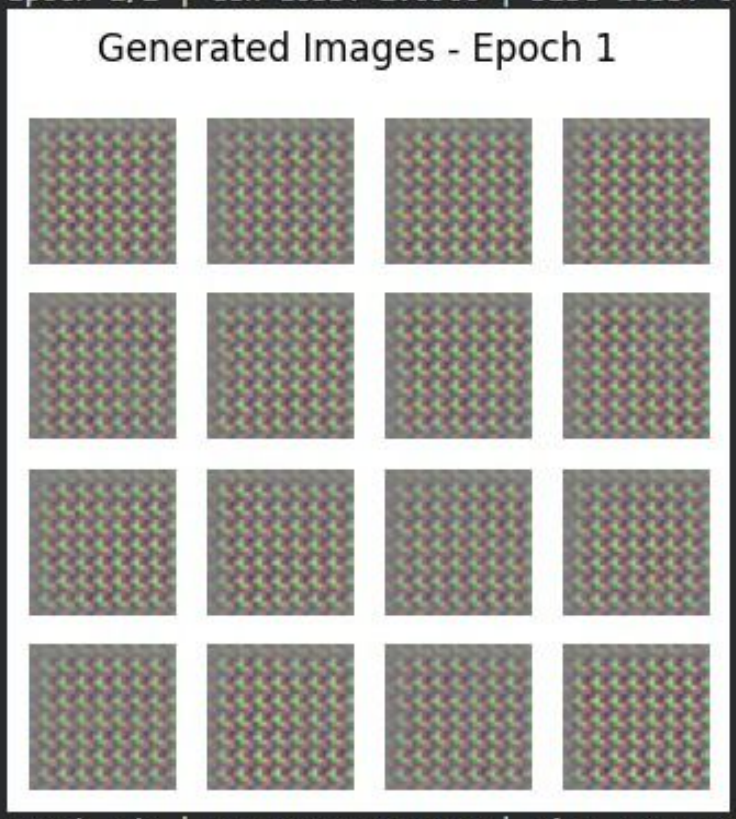
Epoch vs loss (GAN)



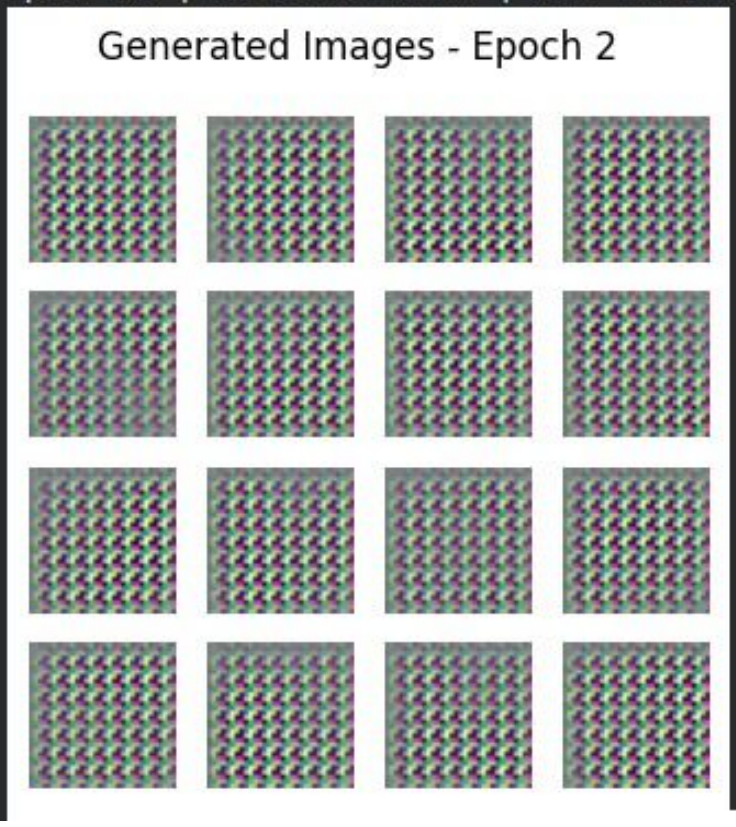
```python
# ✅ Lab 12 (Fast Version): DCGAN to Generate Color Images (CIFAR-10)
import tensorflow as tf
from tensorflow.keras import layers
import matplotlib.pyplot as plt
import numpy as np
import time
(x_train, _), (_, _) = tf.keras.datasets.cifar10.load_data()
x_train = (x_train.astype("float32") - 127.5) / 127.5  # normalize to [-1,1]
x_train = x_train[:5000]  # only 5k images for faster training

BUFFER_SIZE = 5000
BATCH_SIZE = 32
train_dataset = tf.data.Dataset.from_tensor_slices(x_train).shuffle(BUFFER_SIZE).batch(BATCH_SIZE)

def build_generator():
    model = tf.keras.Sequential([
        layers.Input(shape=(100,)),
        layers.Dense(8*8*128, use_bias=False),
        layers.BatchNormalization(),
        layers.LeakyReLU(),
        layers.Reshape((8, 8, 128)),
        layers.Conv2DTranspose(64, (5,5), strides=(2,2), padding="same", use_bias=False),
        layers.BatchNormalization(),
        layers.LeakyReLU(),
        layers.Conv2DTranspose(3, (5,5), strides=(2,2), padding="same", use_bias=False, activation="tanh"),
    ])
    return model

def build_discriminator():
    model = tf.keras.Sequential([
        layers.Input(shape=(32, 32, 3)),
        layers.Conv2D(64, (5,5), strides=(2,2), padding="same"),
        layers.LeakyReLU(),
        layers.Dropout(0.3),
        layers.Conv2D(128, (5,5), strides=(2,2), padding="same"),
        layers.LeakyReLU(),
        layers.Dropout(0.3),
        layers.Flatten(),
        layers.Dense(1)
    ])
    return model

generator = build_generator()
discriminator = build_discriminator()

cross_entropy = tf.keras.losses.BinaryCrossentropy(from_logits=True)
gen_optimizer = tf.keras.optimizers.Adam(1e-4)
disc_optimizer = tf.keras.optimizers.Adam(1e-4)

@tf.function
def train_step(images):
    noise = tf.random.normal([BATCH_SIZE, 100])
    with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
        generated_images = generator(noise, training=True)
        real_output = discriminator(images, training=True)
        fake_output = discriminator(generated_images, training=True)
        gen_loss = cross_entropy(tf.ones_like(fake_output), fake_output)
        disc_loss = (
            cross_entropy(tf.ones_like(real_output), real_output)
            + cross_entropy(tf.zeros_like(fake_output), fake_output)
        )
```

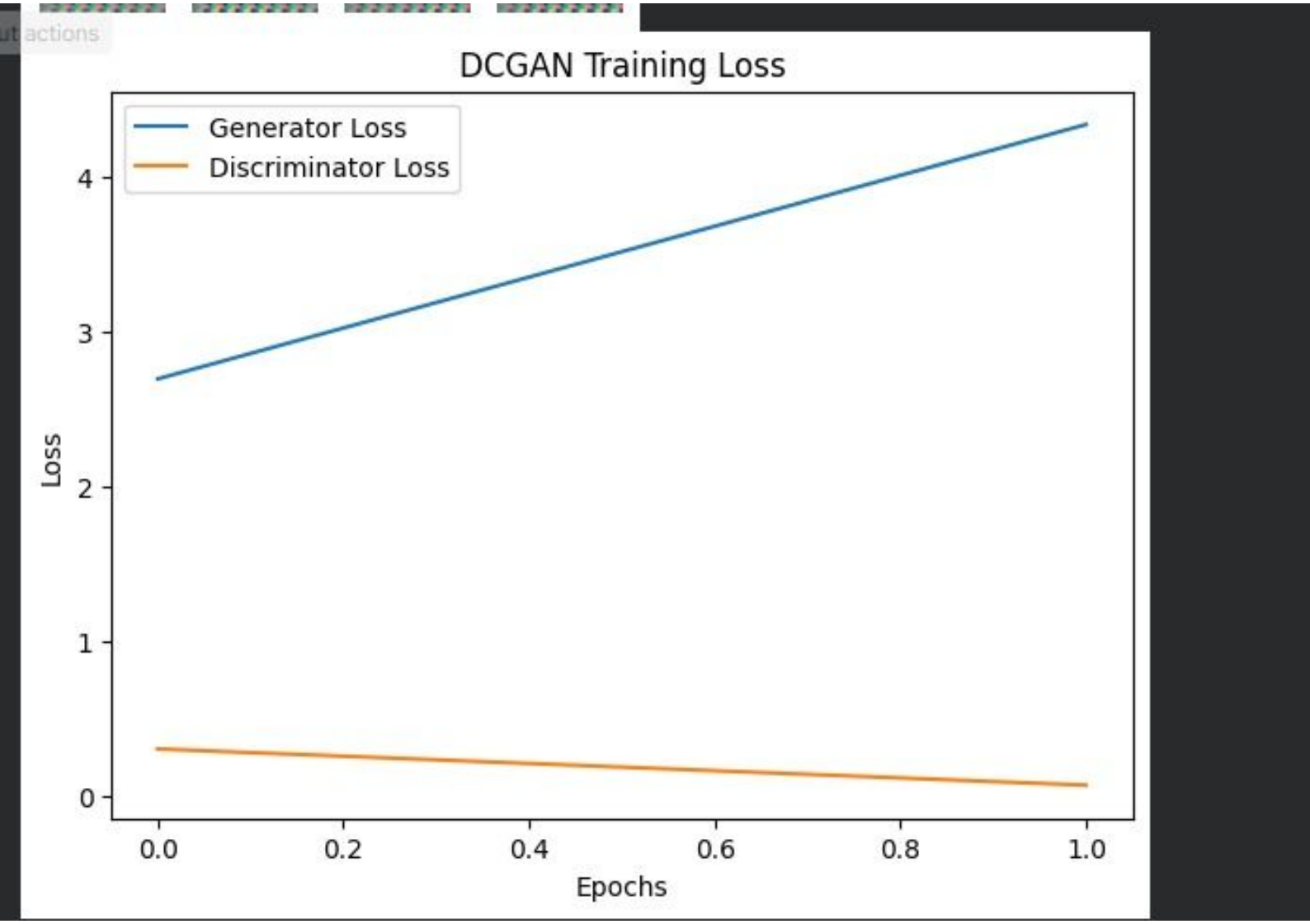
```
            data = data[0]
        z_mean, z_log_var, z = self.encoder(data)
        reconstruction = self.decoder(z)
        reconstruction_loss = tf.reduce_mean(
            tf.keras.losses.binary_crossentropy(data, reconstruction) * 784
        )
        kl_loss = -0.5 * tf.reduce_mean(
            1 + z_log_var - tf.square(z_mean) - tf.exp(z_log_var)
        )
        total_loss = reconstruction_loss + kl_loss
        return {"loss": total_loss}

# 5 Compile and Train
vae = VAE(encoder, decoder)
vae.compile(optimizer=tf.keras.optimizers.Adam())
history = vae.fit(
    x_train,
    x_train,
    epochs=10,
    batch_size=256,
    validation_data=(x_test, x_test),
)

# 6 Plot Training vs Validation Loss
plt.plot(history.history["loss"], label="Training Loss")
plt.plot(history.history["val_loss"], label="Validation Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.title("VAE Training vs Validation Loss")
plt.show()

# 7 Reconstruct Test Images
decoded_imgs = vae.decoder(vae.encoder(x_test)[2]).numpy()

n = 10
plt.figure(figsize=(20, 4))
for i in range(n):
    # Original
    ax = plt.subplot(2, n, i + 1)
    plt.imshow(x_test[i].reshape(28, 28), cmap="gray")
    plt.axis("off")
    # Reconstructed
    ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(decoded_imgs[i].reshape(28, 28), cmap="gray")
    plt.axis("off")
plt.suptitle("Top: Original Images | Bottom: VAE Reconstructions")
plt.show()
```

```
Epoch 1/2 | Gen Loss: 2.6906 | Disc Loss: 0.2989 | Time: 36.21s
```




```
Epoch 2/2 | Gen Loss: 4.3347 | Disc Loss: 0.0647 | Time: 29.22s
```




DCGAN Training Loss
Generator Loss
Discriminator Loss
Loss
Epochs
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0